A Modal Logic for the Decision-Theoretic Projection Problem

Gavin Rens[1], Thomas Meyer[1] and Gerhard Lakemeyer[2]

[1]*Centre for Artificial Intelligence Research, University of KwaZulu-Natal, and CSIR Meraka, South Africa.*

[2]*RWTH Aachen University, Germany.*

{*grens, tmeyer*}*@csir.co.za, gerhard@cs.rwth-aachen.de*

Keywords: Logic, POMDP, Projection, Decision-theory

Abstract: We present a decidable logic in which queries can be posed about (i) the degree of belief in a propositional sentence after an arbitrary finite number of actions and observations and (ii) the utility of a finite sequence of actions after a number of actions and observations. Another contribution of this work is that a POMDP model specification is allowed to be partial or incomplete with no restriction on the lack of information specified for the model. The model may even contain information about non-initial beliefs. Essentially, entailment of arbitrary queries (expressible in the language) can be answered. A sound, complete and terminating decision procedure is provided.

1 INTRODUCTION

Symbolic logic is good for representing information compactly and it is good for reasoning with that information. However, only in the last two or three decades has research gone into developing ways to employ logic for representing stochastic information. One formalism for modelling agents in stochastic domains and for determining 'good' sequences of actions is the *partially observable Markov decision process* (POMDP) (Smallwood and Sondik, 1973; Monahan, 1982). The popularity of the POMDP approach is, arguably, due to its relative simplicity and intuitiveness, and its general applicability to a wide range of stochastic domains. In this paper, we propose the *Stochastic Decision Logic* (SDL), a modal logic with a POMDP semantics. It combines the benefits of POMDP theory and logic for posing entailment queries about stochastic domains.

In POMDPs, actions have nondeterministic results and observations are uncertain. In other words, the effect of some chosen action is somewhat unpredictable, yet may be predicted with a probability of occurrence, and the world is not directly observable: some data are observable and the agent infers how likely it is that the world is in some particular state. The agent may thus believe to some degree—for each possible state—that it is in that state, but it is never certain exactly which state it is in. In fact, the agent typically maintains a probability distribution over the states reflecting its conviction for being in a state, for each state.

Traditionally, to make any deductions in POMDP theory, a domain model must be completely specified. Another contribution of this work is that it allows the user to determine whether or not a set of sentences is entailed by an arbitrarily precise specification of a POMDP model. By "arbitrarily precise specification" we mean that the transition function, the perception function, the reward function or the initial belief-state might not be completely defined by the logical specification provided. Another view is that the logic allows for the (precise) specification of and reasoning over classes of POMDP models.

This work is not meant to be a logic-based version of all POMDP theroy; it is meant to be a logic with POMDP semantics for online reasoning in stochastic domains.

Full-scale planning will not be considered here. However, as a preliminary step, projections concerning epistemic situations and expected rewards will be possible. That is, at this stage, we have not developed a procedure to produce a reward-maximizing policy conditioned on observations. There is, however, a procedure to determine whether some hypothesised situation follows from a knowledge base of the system and some beliefs about the system state. More precisely, with the SDL, an agent can (i) determine the degree of belief in a propositional sentence after an arbitrary finite number of actions and observations and (ii) the utility[1] of a finite sequence of actions after a number of actions and observations.

Imagine a robot that is in need of an oil refill. There is an open can of oil on the floor within reach of its gripper. If there is nothing else in the robot's gripper, it can grab the can (or miss it, or knock it over) and it can drink the oil by lifting the can to its 'mouth' and pouring the contents in (or miss its mouth and spill). The robot may also want to confirm whether there is anything left in the oil-can by weighing its contents with its 'weight' sensor. And once holding the can, the robot may wish to replace it on the floor. There are also rewards and costs involved, which are explained in the Examples section of the paper. The domain is (partially) formalized as follows. The robot has the set of (intended) actions $\mathcal{A} = \{\texttt{grab}, \texttt{drink}, \texttt{weigh}, \texttt{replace}\}$ with expected intuitive meanings. The robot can perceive observations only from the set $\Omega = \{\texttt{Nil}, \texttt{Light}, \texttt{Medium}, \texttt{Heavy}\}$. Intuitively, when the robot performs a `weigh` action (i.e., it activates its 'weight' sensor) it will perceive either `Light`, `Medium` or `Heavy`; for other actions, it will perceive `Nil`. The robot experiences its world (domain) through two Boolean features: $\mathcal{F} = \{\texttt{full}, \texttt{holding}\}$ meaning that the robot believes the oil-can is full and respectively that it is currently holding something in its gripper.

In the following informal examples, several syntactic elements are mentioned which are formally defined in Section 2.1. $\mathbf{B}\varphi \geq p$ is read 'The degree of belief in φ is greater than or equal to p'. $\mathbf{U}\Lambda > r$ is read 'The utility of performing action sequence Λ is greater than r'. Given a complete formalization $\mathcal{K}$ of the scenario sketched here, a robot may have the following queries:

- Is the degree of belief that I'll have the oil-can in my gripper greater than or equal to 0.9, after I attempt grabbing it twice in a row? That is, does $[\![\texttt{grab} + \texttt{obsNil}]\!]\, [\![\texttt{grab} + \texttt{obsNil}]\!]\mathbf{B}\,\texttt{holding} \geq 0.9$ follow from $\mathcal{K}$?
- After grabbing the can, then perceiving that it has medium weight, is the utility of drinking the contents of the oil-can, then placing it on the floor, more than 6 units? That is, does $[\![\texttt{grab} + \texttt{obsNil}]\!]$ $[\![\texttt{weigh} + \texttt{obsMedium}]\!]\ \mathbf{U}[\![\texttt{drink}]\!][\![\texttt{replace}]\!] > 6$ follow from $\mathcal{K}$?

Related Work: Recently, some researchers have investigated formal languages for compactly representing POMDPs (Boutilier and Poole, 1996; Geffner and Bonet, 1998; Hansen and Feng, 2000; Wang and Schmolze, 2005; Sanner and Kersting, 2010; Lison, 2010; Wang and Khardon, 2010). They also mention that with a logical language for specifying models, decision-making algorithms can exploit the structure found in these logical specifications. They are not presented as *logics*, though, and logical theorem proving is thus not possible for them.

[1] By "utility", we mean 'expected rewards'.

De Weerdt et al. (1999) present a modal logic to deal with imprecision in robot actions and sensors. Their models do not contain an accessibility relation, which makes it hard to understand what it means for an action to be executed. They cannot deal with utilities of actions, and no system for determining truth of statements is provided.

Bacchus et al. (1999) supply a theory for reasoning with noisy sensors and effectors, with graded belief. They use the situation calculus (McCarthy, 1963) to specify their approach but some elements fall outside the logical language. They don't address utilities of actions.

$\mathcal{ESP}$ (Gabaldon and Lakemeyer, 2007) is a construction of Bacchus et al.'s approach with some improvements. It is founded on $\mathcal{ES}$ (Levesque and Lakemeyer, 2004), which is a fragment of the situation calculus. The semantics of SDL is arguably simpler than that of $\mathcal{ESP}$, because SDL fixes its semantics on POMDPs. In the long-run, this may be a disadvantage of the SDL, though. With any logic based on the situation calculus or first-order logic, decidability of entailment comes into question. The SDL's entailment procedure is decidable.

In Poole (1998)'s Independent Choice Logic using the situation calculus (ICL_{SC}): "The representation in this paper can be seen as a representation for POMDPs". Belief-states can be expressed and belief update can be performed (but maintenance of belief-states is not a necessary component of the system). Even programs that are sequences of actions conditioned on observations can be expressed for agents to adopt. The ICL_{SC} is a relatively rich framework, with acyclic logic programs which may contain variables, quantification and function symbols. For certain applications, the SDL may be preferred due to its comparative simplicity, and it may be easier to understand by people familiar with POMDPs. Finally, decidability of inferences made in the ICL_{SC} are, in general, not guaranteed.

Iocchi et al. (2009) present a logic called $\mathcal{E}+$ for reasoning about agents with sensing, qualitative nondeterminism and probabilistic uncertainty in action outcomes. Planning with sensing and uncertain actions is also dealt with. The application area is plan generation for agents with nondeterministic and probabilistic uncertainty. Noisy sensing is not dealt with, that is, sensing actions are deterministic. They mention that although they would like to be able to represent action rewards and costs as in POMDPs, $\mathcal{E}+$ does not yet provide the facilities.

PRISM is a framework for model-checking representations of systems with a probabilistic character (Kwiatkowska et al., 2010). Kwiatkowska et al. (2010) show how MDPs can be represented with an extension of Probabilistic Computation Tree Logic (Hansson and Jonsson, 1994). PRISM can then determine whether the occurrence of some event satisfies a given probability bound. To our knowledge, PRISM has not been extended to represent POMDPs. Moreover, by definition, model-checking requires full specification of a system. However, we could learn something from the implementation of PRISM (www.prismmodelchecker.org) for the future development of the SDL, or PRISM could be extended with ideas from the SDL.

There is another sense in which an incomplete model can be dealt with; it can be learnt. Ross et al. (2011) outline the Bayes-Adaptive POMDP framework to reinforcement learning, which allows them to "explicitly target the exploration-exploitation problem in a coherent mathematical framework." Our work is different in that we do not tackle the learning problem; our work suggests a way for an agent to make decisions with incomplete models without considering whether its actions will also help it explore wisely. There are problems for which an agent should explore its environment while working on its task. But there may also be problems for which the agent should not explore (anymore?) and simply work on the task at hand with the given information (domain model).

When it comes to the projection task (in the first-order setting), work by Shirazi and Amir (2011) concerning "filtering" in the incremental update of the belief-state, may be important to look at.

Next, our logic is defined. Then in Section 3, we describe a decision procedure for checking entailment queries. In Section 4, a framework for domain specification is described and some examples of the logic in use are provided.

2 The Stochastic Decision Logic

The SDL's foundations are in the Specification Logic of Actions with Probability (Rens et al., 2014b) and the Specification Logic of Actions and Observations with Probability (Rens et al., 2014a).

2.1 Syntax

The syntax is very carefully designed to provide the required expressiveness, and no more.

The vocabulary of our language contains six sorts of objects:

1. a finite set of *fluents* $\mathcal{F} = \{f_1, \ldots, f_n\}$,
2. a finite set of names of atomic *actions* $\mathcal{A} = \{\alpha_1, \ldots, \alpha_n\}$,
3. a countable set of *action variables* $V_{\mathcal{A}} = \{v_1^a, v_2^a, \ldots\}$,
4. a finite set of names of atomic *observations* $\Omega = \{\varsigma_1, \ldots, \varsigma_n\}$,
5. a countable set of *observation variables* $V_\Omega = \{v_1^o, v_2^o, \ldots\}$.
6. all *real numbers* $\mathbb{R}$,

We refer to elements of $\mathcal{A} \cup \Omega$ as *constants*. We work in a multi-modal setting, in which we have modal operators $[\alpha]$, one for each $\alpha \in \mathcal{A}$. And $[\![\alpha + \varsigma]\!]$ is a *belief update operator* (or *update operator* for short). Intuitively, $[\![\alpha + \varsigma]\!]\Theta$ means 'Θ holds in the belief-state resulting from performing action α and then perceiving observation ς'. For instance, $[\![\alpha_1 + \varsigma_1]\!]\,[\![\alpha_2 + \varsigma_2]\!]$ expresses that the agent executes α_1 then perceives ς_1 then executes α_2 then perceives ς_2. **B** is a modal operator for belief and **U** is a modal operator for utility.

We first define a language $\mathcal{L}$, then a useful sublanguage $\mathcal{L}_{SDL} \subset \mathcal{L}$. The reason why we define $\mathcal{L}$ is because it is easier to define the truth conditions for $\mathcal{L}$; the truth conditions for $\mathcal{L}_{SDL}$ then follow directly.

Definition 2.1. *First the propositional fragment:* $\varphi ::= f \mid \top \mid \neg\varphi \mid \varphi \wedge \varphi$, *where* $f \in \mathcal{F}$.

Then the fragment Φ *used in formulae of the form* $\varphi \Rightarrow \Phi$ *(see the definition of* Θ *below).*

Let $\alpha \in (V_{\mathcal{A}} \cup \mathcal{A})$, $v^a \in V_{\mathcal{A}}$, $\varsigma \in (V_\Omega \cup \Omega)$, $v^o \in V_\Omega$, $p \in [0,1]$, $r \in \mathbb{R}$ *and* $\bowtie \in \{<, \leq, =, \geq, >\}$.[2]

$$\Phi ::= \varphi \mid \alpha = \alpha \mid \varsigma = \varsigma \mid Reward(r) \mid Cost(\alpha, r) \mid [\alpha]\varphi \bowtie p \mid (\varsigma|\alpha) \bowtie p \mid (\forall v^a)\Phi \mid (\forall v^o)\Phi \mid \neg\Phi \mid \Phi \wedge \Phi.$$

where φ *is defined above.*

$[\alpha]\varphi \bowtie p$ *is read 'The probability x of reaching a* φ*-world after executing* α *is such that* $x \bowtie p$*'. Whereas* $[\alpha]$ *is a modal operator,* $(\varsigma|\alpha)$ *is a predicate;* $(\varsigma|\alpha) \bowtie p$ *is read 'The probability x of perceiving* ς*, given* α *was performed is such that* $x \bowtie p$*'.*

The language of $\mathcal{L}$ *is defined as* Θ*:*

$$\Lambda ::= [\![\alpha]\!] \mid \Lambda[\![\alpha]\!]$$

$$\Theta ::= \top \mid \alpha = \alpha \mid \varsigma = \varsigma \mid Cont(\alpha, \varsigma) \mid \mathbf{B}\varphi \bowtie p \mid \mathbf{U}\Lambda \bowtie r \mid \varphi \Rightarrow \Phi \mid [\![\alpha + \varsigma]\!]\Theta \mid (\forall v^a)\Theta \mid (\forall v^o)\Theta \mid \neg\Theta \mid \Theta \wedge \Theta \mid \Theta \vee \Theta,$$

[2] $[0,1]$ denotes $\mathbb{R} \cap [0,1]$.

where φ and Φ are defined above.

The scope of quantifier $(\forall v')$ is determined in the same way as is done in first-order logic. A variable v appearing in a formula Θ is said to be bound by quantifier $(\forall v')$ if and only if v is the same variable as v' and is in the scope of $(\forall v')$. If a variable is not bound by any quantifier, it is free. In $\mathcal{L}$, variables are not allowed to be free; they are always bound.

$Cont(\alpha, \varsigma)$ is read 'Consciousness continues after executing α and then perceiving ς'. $\mathbf{B}\varphi \bowtie p$ is read 'The degree of belief x in φ is such that $x \bowtie p$'. Performing $\Lambda = [\![\alpha_1]\!][\![\alpha_2]\!]\cdots[\![\alpha_z]\!]$ means that α_1 is performed, then α_2 then ... then α_z. $\mathbf{U}\Lambda \bowtie r$ is thus read 'The utility x of performing Λ is such that $x \bowtie r$'. Evaluating some sentence Ψ after a sequence of z update operations, means that Ψ will be evaluated after the agent's belief-state has been updated according to the sequence

$$\underbrace{[\![\alpha+\varsigma]\!]\cdots[\![\alpha'+\varsigma']\!]}_{z \text{ times}}$$

of actions and observations. $\varphi \Rightarrow \Phi$ is read 'It is a general law of the domain that Φ holds in all situations (worlds) which satisfy φ'.

Definition 2.2. *The language of SDL, denoted $\mathcal{L}_{SDL}$, is the subset of formulae of $\mathcal{L}$ excluding formulae containing subformulae of the form $\neg(\varphi \Rightarrow \Phi)$.*

For instance, sentences of the form $\neg(\varphi \Rightarrow \Phi) \wedge (\varphi' \Rightarrow \Phi') \wedge \Theta \notin \mathcal{L}_{SDL}$, but $(\varphi \Rightarrow \Phi) \wedge (\varphi' \Rightarrow \Phi') \wedge \Theta \in \mathcal{L}_{SDL}$. And, for instance, $\neg(\forall v')(\varphi \Rightarrow \Phi) \vee (\varphi' \Rightarrow \Phi') \vee \Theta \notin \mathcal{L}_{SDL}$, but $(\forall v')(\varphi \Rightarrow \Phi) \vee (\varphi' \Rightarrow \Phi') \vee \Theta \in \mathcal{L}_{SDL}$. The reason why $\mathcal{L}_{SDL}$ is defined to exclude $\neg(\varphi \Rightarrow \Phi)$ is because such sentences cause unnecessary technical difficulties in the decision procedure. Rens's doctoral thesis (Rens, 2014, Chap. 8) containing a detailed explanation.

$\bot$ abbreviates $\neg\top$, $\theta \rightarrow \theta'$ abbreviates $\neg\theta \vee \theta'$ and $\leftrightarrow$ abbreviates $(\theta \rightarrow \theta') \wedge (\theta' \rightarrow \theta)$. In grammars φ and Φ, $\phi \vee \phi'$ abbreviates $\neg(\neg\phi \wedge \neg\phi')$, but in grammar Θ, $\vee$ is defined directly, because otherwise its definition in terms of $\neg$ and $\wedge$ would involve formulas of the form $\neg(\varphi \Rightarrow \Phi)$, which are precluded in $\mathcal{L}_{SDL}$. $\rightarrow$ and $\leftrightarrow$ have the weakest bindings, with $\Rightarrow$ just stronger; and $\neg$ the strongest. Parentheses enforce or clarify the scope of operators conventionally.

$c = c'$ is an equality literal, $Reward(r)$ is a reward literal, $Cost(\alpha, r)$ is a cost literal, $[\alpha]\varphi \bowtie p$ is a dynamic literal, $(\varsigma|\alpha) \bowtie p$ is a perception literal, and $\varphi \Rightarrow \Phi$ is a law literal. $Cont(\alpha, \varsigma)$ is a continuity literal, $\mathbf{B}\varphi \bowtie p$ is a belief literal and $\mathbf{U}\Lambda \bowtie r$ is a utility literal. The negation of all these literals are also literals with the associated names.

2.2 Semantics

Formally, a partially observable Markov decision process (POMDP) is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \mathcal{Z}, \mathcal{P}, b^0 \rangle$: a finite set of states $\mathcal{S} = \{s_1, s_2, \ldots, s_n\}$; a finite set of actions $\mathcal{A} = \{a_1, a_2, \ldots, a_k\}$; the *state-transition function*, where $\mathcal{T}(s, a, s')$ is the probability of being in s' after performing action a in state s; the *reward function*, where $\mathcal{R}(a, s)$ is the reward gained for executing a while in state s; a finite set of observations $\mathcal{Z} = \{z_1, z_2, \ldots, z_m\}$; the *observation function*, where $\mathcal{P}(s', a, z)$ is the probability of observing z in state s' resulting from performing action a in some other state; and b^0 is the initial probability distribution over all states in $\mathcal{S}$.

Let b be a total function from $\mathcal{S}$ into $\mathbb{R}$. Each state s is associated with a probability $b(s) = p \in \mathbb{R}$, such that b is a probability distribution over the set $\mathcal{S}$ of all states. b can be called a *belief-state*.

An important function in POMDP theory is the function that updates the agent's belief-state, or the *state estimation* function SE. $SE(a, z, b) = b_n$, where $b_n(s')$ is the probability of the agent being in state s' in the 'new' belief-state b_n, relative to a, z and the 'old' belief-state b. Notice that $SE(\cdot)$ requires a belief-state, an action and an observation as inputs to determine the new belief-state.

When the states an agent can be in are *belief-states* (as opposed to objective, single states in $\mathcal{S}$), the reward function $\mathcal{R}$ must be lifted to operate over belief-states. The *expected* reward $\rho(a, b)$ for performing an action a in a belief-state b is defined as $\sum_{s \in \mathcal{S}} \mathcal{R}(a, s) b(s)$.

Let $w : \mathcal{F} \rightarrow \{0, 1\}$ be a total function that assigns a truth value to each fluent. We call w a *world*. Let C be the set of $2^{|\mathcal{F}|}$ *conceivable worlds*, that is, all possible functions w.

Definition 2.3. *An SDL structure is a tuple $\mathcal{D} = \langle T, P, U \rangle$ such that*

- *$T : \mathcal{A} \rightarrow \{T_\alpha \mid \alpha \in \mathcal{A}\}$, where $T_\alpha : (C \times C) \rightarrow [0, 1]$ is a total function from pairs of worlds into the reals. That is, T provides a transition (accessibility) relation T_α for each action in $\mathcal{A}$. For every $w^- \in C$, it is required that either $\sum_{w^+ \in C} T_\alpha(w^-, w^+) = 1$ or $\sum_{w^+ \in C} T_\alpha(w^-, w^+) = 0$.*[3]
- *$P : \mathcal{A} \rightarrow \{P_\alpha \mid \alpha \in \mathcal{A}\}$, where $P_\alpha : (C \times \Omega) \rightarrow [0, 1]$ is a total function from pairs in $C \times \Omega$ into the reals. That is, P provides a perceivability relation*

[3]Either the action is executable and there is a probability distribution (the summation is 1) or the action is inexecutable (the summation is 0). Letting the sum equal a number not 1 or 0 would lead to badly defined semantics.

P_α for each action in $\mathcal{A}$. For all $w^+ \in C$, if there exists a $w^- \in C$ such that $T_\alpha(w^-, w^+) > 0$, then $\sum_{\varsigma\in\Omega} P_\alpha(w^+, \varsigma) = 1$, else $\sum_{\varsigma\in\Omega} P_\alpha(w^+, \varsigma) = 0$;

- *U is a pair $\langle Re, Co\rangle$, where $Re : C \to \mathbb{R}$ is a reward function and Co is a mapping that provides a cost function $Co_\alpha : C \to \mathbb{R}$ for each $\alpha \in \mathcal{A}$.*

As in POMDPs, in the SDL, an agent typically does not know in which world $w \in C$ it actually is, but for each w it has a degree of belief that it is in that world. From now on, let $b : C \to [0, 1]$ be a probability distribution over C, still referred to as a *belief-state*. The degree of belief in w is denoted by the probability measure $b(w)$.

Definition 2.4. *The probability of reaching the next belief-state b' from the current belief-state b, given α and ς, is $Pr_{NB}(\alpha, \varsigma, b) = \sum_{w'\in C} P_\alpha(\varsigma, w') \sum_{w\in C} T_\alpha(w, w')b(w)$.*

The above definition is from standard POMDP theory.

Definition 2.5. *We define a* belief update *function $BU(\alpha, \varsigma, b) = b'$:*

$$b'(w') = \frac{P_\alpha(w', \varsigma) \sum_{w\in C} T_\alpha(w, w')b(w)}{Pr_{NB}(\alpha, \varsigma, b)},$$

for $Pr_{NB}(\alpha, \varsigma, b) \neq 0$.

$BU(\cdot)$ has the same intuitive meaning as the state estimation function $SE(\cdot)$ of POMDP theory.

Given the opportunity to be slightly more clear about the specification of rewards in the SDL, we interpret $\mathcal{R}(a, s)$ of POMDPs as $R(s) - C(a, s)$, where $R(s)$ provides the positive reward portion of $R(a, s)$ and $C(a, s)$ provides the punishment or cost portion. By this interpretation, we assume that simply being in a state has an intrinsic reward (independent of an action), however, that punishment is conditional on actions and the states in which they are executed. There are many other ways to interpret $\mathcal{R}(a, s)$, and $\mathcal{R}(a, s)$ is not even the most general reward function possible; a more general function is $\mathcal{R}(s, a, s')$ meaning that rewards depend on a state s, an action executed in s and a state s' reached due to performing a in s. The SDL adopts one of several reasonable approaches. In the semantics of the SDL, we equate a state s with a world w and an action a as $\alpha \in \mathcal{A}$, and interpret $\mathcal{R}(a, s)$ as $Re(w) - Co_\alpha(w)$. We derive a reward function over belief-states for the SDL in a similar fashion as we did with $\rho(a, b)$ of POMDP theory, however, including the notion of cost: $RC(\alpha, b) = \sum_{w\in C}(Re(w) - Co_\alpha(w))b(w)$.

Let $\alpha, \alpha' \in \mathcal{A}$, $\varsigma, \varsigma' \in \Omega$, $p \in [0, 1]$ and $r \in \mathbb{R}$. Let $f \in \mathcal{F}$ and let Θ be any sentence in $\mathcal{L}$. Let $\bowtie \in \{<, \leq, =, \geq, >\}$. If $\Theta \in \mathcal{L}$ is *satisfied* at world w and belief-state b in SDL structure $\mathcal{D}$, we write $\mathcal{D}bw \models \Theta$. Some of the conditions for satisfaction are reproduced below.

$\mathcal{D}bw \models \alpha = \alpha' \iff \alpha$ and α' are the same element;

$\mathcal{D}bw \models \varsigma = \varsigma' \iff \varsigma$ and ς' are the same element;

$\mathcal{D}bw \models Reward(r) \iff Re(w) = r$;

$\mathcal{D}bw \models Cost(\alpha, c) \iff Co_\alpha(w) = c$;

$\mathcal{D}bw \models [\alpha]\varphi \bowtie p \iff \sum_{\substack{w'\in C\\ \mathcal{D}bw'\models\varphi}} T_\alpha(w, w') \bowtie p$;

$\mathcal{D}bw \models (\varsigma|\alpha) \bowtie p \iff P_\alpha(w, \varsigma) \bowtie p$;

$\mathcal{D}bw \models Cont(\alpha, \varsigma) \iff Pr_{NB}(\alpha, \varsigma, b) \neq 0$;

$\mathcal{D}bw \models \mathbf{B}\varphi \bowtie p \iff \sum_{\substack{w'\in C\\ \mathcal{D}bw'\models\varphi}} b(w') \bowtie p$;

$\mathcal{D}bw \models \mathbf{U}[\![\alpha]\!] \bowtie r \iff RC(\alpha, b) \bowtie r$;

$\mathcal{D}bw \models \mathbf{U}[\![\alpha]\!]\Lambda \bowtie r \iff$
$\Big(RC(\alpha, b) + \sum_{\varsigma\in\Omega} Pr_{NB}(\alpha, \varsigma, b) \cdot r'\Big) \bowtie r$,
where $\mathcal{D}b'w \models \mathbf{U}\Lambda = r'$ for $b' = BU(\alpha, \varsigma, b)$;

$\mathcal{D}bw \models \varphi \Rightarrow \Theta \iff$
for all $w' \in C$, if $\mathcal{D}bw' \models \varphi$ then $\mathcal{D}bw' \models \Theta$;

$\mathcal{D}bw \models [\![\alpha + \varsigma]\!]\Theta \iff Pr_{NB}(\alpha, \varsigma, b) \neq 0$ and $\mathcal{D}b'w \models \Theta$, where $b' = BU(\alpha, \varsigma, b)$;

$\mathcal{D}bw \models (\forall v^a)\Upsilon \iff \mathcal{D}bw \models \Upsilon|^{v^a}_{\alpha_1} \wedge \ldots \wedge \Upsilon|^{v^a}_{\alpha_n}$;

$\mathcal{D}bw \models (\forall v^o)\Upsilon \iff \mathcal{D}bw \models \Upsilon|^{v^o}_{\varsigma_1} \wedge \ldots \wedge \Upsilon|^{v^o}_{\varsigma_n}$,

where Υ is a formula from the grammar Φ or Θ, and we write $\Upsilon|^v_c$ to mean the formula Υ with all occurrences of variables $v \in (V_{\mathcal{A}} \cup V_\Omega)$ appearing in it replaced by constant $c \in \mathcal{A} \cup \Omega$ of the right sort.

A sentence $\Theta \in \mathcal{L}$ is *satisfiable* if there exists a structure $\mathcal{D}$, a belief-state b and a world w such that $\mathcal{D}bw \models \Theta$, else Θ is *unsatisfiable*. Let $\mathcal{K} \subset \mathcal{L}$. We say that $\mathcal{K}$ entails Θ (denoted $\mathcal{K} \models \Theta$) if for all structures $\mathcal{D}$, all belief-states b, all $w \in C$: if $\mathcal{D}bw \models \kappa$ for every $\kappa \in \mathcal{K}$, then $\mathcal{D}bw \models \Theta$. When $\mathcal{K}$ is a finite subset of $\mathcal{L}_{SDL}$ and $\Psi \in \mathcal{L}_{SDL}$, it is easy to show that $\mathcal{K} \models \Psi \iff \bigwedge_{\kappa\in\mathcal{K}} \kappa \wedge \neg\Psi$ is unsatisfiable. The SDL decision procedure for entailment is based on this latter correspondence.

3 The Decision Procedure for SDL Entailment

Informally, a query is satisfiable if there exists a way of filling in missing domain information about rewards, transitions, perceptions, etcetera, so that the query is true. And a query should be valid if all ways of extending the supplied model information makes the query true.

We provide a sketch of the (formal) decision procedure for checking whether entailments of the form $\mathcal{K} \models \Psi$ hold. Our strategy is to set up a tableau tree for $\bigwedge_{\kappa\in\mathcal{K}} \kappa \wedge \neg\Psi$, and then check whether or not every leaf node of the tree after full expansion implies a contradiction. If every leaf node does imply a contradiction, then $\bigwedge_{\kappa \in \mathcal{K}} \kappa \wedge \neg \Psi$ is unsatisfiable and $\mathcal{K} \models \Psi$ holds.

There are two phases in the decision procedure. The first phase uses a tableau approach to (i) catch 'traditional' contradictions, (ii) separate formulae into literals and (iii) prepare the literals for processing in the second phases. We shall call this the *tableau* phase. The second phase creates systems of inequalities, checking their feasibility. We shall call this the *systems of inequalities* (SI) phase.

An *activity sequence* is either 0 or a sequence of the form $0 \xrightarrow{\alpha_1,\varsigma_1} e_1 \xrightarrow{\alpha_2,\varsigma_2} e_2 \cdots \xrightarrow{\alpha_z,\varsigma_z} e_z$. Intuitively, an activity sequence represents a hypothetical sequence of actions and associated perceptions. The e_i represent belief-states; e_z is an integer which uniquely identifies the belief-state reached after the occurrence of the sequence $\alpha_1, \varsigma_1, \alpha_2, \varsigma_2, \cdots \alpha_z, \varsigma_z$ of actions and observations. The e_i are called *activity points*—because they represent an agent's state of mind at some point after a sequence of activities.

In the following discussion, and also later, we employ some abbreviations: The set of fluents $\mathcal{F} = \{\texttt{full}, \texttt{holding}\}$ is abbreviated to $\{f, h\}$. The set of actions $\mathcal{A} = \{\texttt{grab}, \texttt{drink}, \texttt{weigh}\}$ is abbreviated to $\{g, d, w\}$. The set of observations $\Omega = \{\texttt{Nil}, \texttt{Light}, \texttt{Medium}, \texttt{Heavy}\}$ is abbreviated to $\{N, L, M, H\}$.

Given some initial belief-state, every clause of a sentence specifies a final belief-state/activity point. For instance, $\mathbf{B}(f \wedge h) = 0.35 \wedge \mathbf{B}(f \wedge \neg h) = 0.35 \wedge \mathbf{B}(\neg f \wedge h) = 0.2 \wedge \mathbf{B}(\neg f \wedge \neg h) = 0.1$ specifies the belief-state $\{(w_1, 0.35), (w_2, 0.35), (w_3, 0.2), (w_4, 0.1)\}$, where $w_1 \models f \wedge h, \ldots, w_4 \models \neg f \wedge \neg h$. And $[\![g + N]\!][\![w + M]\!]\mathbf{B}h > 0.85$ specifies belief-state $BU(w, M, BU(g, N, b^0))$, where b^0 is some initial belief-state. Now it is obvious that

$$\begin{aligned}
&\mathbf{B}(f \wedge h) = 0.35 \ \wedge\ \mathbf{B}(f \wedge \neg h) = 0.35\ \wedge \\
&\mathbf{B}(\neg f \wedge h) = 0.2\ \wedge\ \mathbf{B}(\neg f \wedge \neg h) = 0.1 \rightarrow \\
&[\![g + N]\!][\![w + M]\!]\mathbf{B}h > 0.85\ \wedge\ [\![g + N]\!][\![w + M]\!]\mathbf{B}h \leq 0.85
\end{aligned}$$

is a contradiction, because in the belief-state reached after the sequence g, N, w, M, an agent cannot have a degree of belief in h both greater-than and less-than-or-equal-to 0.85. This is a very simple example, but the need for the maintenance of activity sequences and activity points becomes much more apparent when one understands that an activity point plays a part in identifying the variables representing the probabilities of being in the different possible worlds at that point.

3.1 The Tableau Phase

A *labeled formula* is a pair (Σ, Ψ), where $\Psi \in \mathcal{L}_{SDL}$ is any formula, and Σ is an activity sequence. If Σ is $0 \xrightarrow{\alpha_1,\varsigma_1} e_1 \cdots \xrightarrow{\alpha_z,\varsigma_z} e_z$, then the concatenation of Σ and $\xrightarrow{\alpha',\varsigma'} e'$, denoted as $\Sigma \xrightarrow{\alpha',\varsigma'} e'$ is the sequence $0 \xrightarrow{\alpha_1,\varsigma_1} e_1 \cdots \xrightarrow{\alpha_z,\varsigma_z} e_z \xrightarrow{\alpha',\varsigma'} e'$. A *node* Γ is a set of labeled formulae. The initial node to which the tableau rules must be applied, is called the *trunk*. A *tree* T is a set of nodes. A tree must include the trunk and only nodes resulting from the application of *tableau rules* to the trunk and subsequent nodes. If one has a tree with trunk $\{(0, \Psi)\}$, we shall say one has *a tree for* Ψ.

A node Γ is a *leaf* node of tree T if no tableau rule has been applied to Γ in T. A node Γ is *closed* if $(\Sigma, \bot) \in \Gamma$ for any Σ. It is *open* if it is not closed. A tree is closed if all of its leaf nodes are closed, else it is open. A rule may not be applied to (i) a closed leaf node or (ii) a formula to which it has been applied higher in the tree.

Some of the tableau rules follow. Let Γ be a leaf node.

- rule $\wedge$: If Γ contains $(\Sigma, \Psi \wedge \Psi')$ or $(\Sigma, \neg(\Psi \vee \Psi'))$, then create child node $\Gamma' = \Gamma \cup \{(\Sigma, \Psi), (\Sigma, \Psi')\}$, respectively, $\Gamma' = \Gamma \cup \{(\Sigma, \neg\Psi), (\Sigma, \neg\Psi')\}$.
- rule $\vee$: If Γ contains $(\Sigma, \Psi \vee \Psi'))$ or $(\Sigma, \neg(\Psi \wedge \Psi'))$, then create child nodes $\Gamma' = \Gamma \cup \{(\Sigma, \Psi)\}$ and $\Gamma'' = \Gamma \cup \{(\Sigma, \Psi')\}$, respectively, child nodes $\Gamma' = \Gamma \cup \{(\Sigma, \neg\Psi)\}$ and $\Gamma'' = \Gamma \cup \{(\Sigma, \neg\Psi')\}$.
- rule $\Rightarrow \wedge$: If Γ contains $(\Sigma, \varphi \Rightarrow \Phi \wedge \Phi')$, then create child node $\Gamma' = \Gamma \cup \{(\Sigma, \varphi \Rightarrow \Phi), (\Sigma, \varphi \Rightarrow \Phi')\}$.
- rule Ξ: If Γ contains $(\Sigma, [\![\alpha + \varsigma]\!]\Psi)$, then: if Γ contains (Σ', Ψ') such that $\Sigma' = \Sigma \xrightarrow{\alpha,\varsigma} e$, then create node $\Gamma' = \Gamma \cup \{(\Sigma', \Psi)\}$, else create child node $\Gamma' = \Gamma \cup \{(\Sigma \xrightarrow{\alpha,\varsigma} e', \Psi)\}$, where e' is a fresh integer.
- rule $\neg\Xi$: If Γ contains $(\Sigma, \neg[\![\alpha + \varsigma]\!]\Psi)$, then create child node $\Gamma' = \Gamma \cup \{(\Sigma, \neg Cont(\alpha, \varsigma) \vee [\![\alpha + \varsigma]\!]\neg\Psi)\}$.

Definition 3.1. *A branch is* saturated *if and only if every rule that can be applied to its leaf node has been applied. A tree is* saturated *if and only all its branches are saturated.*

3.2 The SI Phase

Let Γ be a leaf node of an open branch of a saturated tree. $SI(\Gamma)$ is the system of inequalities generated from the formulae in Γ (as explained below). After the tableau phase is completed, the SI phase begins. Let T be a saturated tree.

For each open leaf node Γ^j_k of T, do the following. If $SI(\Gamma^j_k)$ is infeasible, then create new leaf node $\Gamma^j_{k+1} = \Gamma^j_k \cup \{(0,\bot)\}$.

Definition 3.2. *A tree is called* finished *after the SI phase is completed.*

Definition 3.3. *If a tree for $\neg\Psi$ is closed, we write $\vdash \Psi$. If there is a finished tree for $\neg\Psi$ with an open branch, we write $\not\vdash \Psi$.*

The generation of $SI(\Gamma)$ from the formulae in Γ is explained in the rest of this section. All variables are assumed implicitly non-negative.

Let $C^\# = \{w_1, w_2, \ldots, w_n\}$ be an ordering of the worlds in C. Let ω^e_k be a variable representing the probability of being in world w_k at activity point e (after a number of activity updates). The equation

$$\omega^0_1 + \omega^0_2 + \cdots + \omega^0_n = 1$$

is in $SI(\Gamma)$ and represents the initial probability distribution over the worlds in C. We may denote an activity sequence as $\Sigma \xrightarrow{\alpha,\varsigma} e$ to refer to the last action α, observation ς and activity point e in the sequence, where Σ may be the empty sequence. We may also denote an activity sequence as Σe to refer only to the last activity point in the sequence; if Σ is the empty sequence, then e is the initial activity point 0.

In the next four subsections, we deal with (i) law literals involving dynamic and perception literals, (ii) activity sequences, (iii) belief literals and (iv) laws involving reward and cost literals, and utility literals.

3.2.1 Action and Perception Laws

For every formulae of the form $(\Sigma, \phi \Rightarrow [\alpha]\varphi \bowtie q) \in \Gamma$ and $(\Sigma, \phi \Rightarrow \neg[\alpha]\varphi \bowtie q) \in \Gamma$, for every j such that $w_j \models \phi$ (where j represents the world in which α is executed),

$$c_1 pr^\alpha_{j,1} + c_2 pr^\alpha_{j,2} + \cdots + c_n pr^\alpha_{j,n} \bowtie q,$$

respectively,

$$c_1 pr^\alpha_{j,1} + c_2 pr^\alpha_{j,2} + \cdots + c_n pr^\alpha_{j,n} \not\bowtie q$$

is in $SI(\Gamma)$, such that $c_k = 1$ if $w_k \models \varphi$, else $c_k = 0$, and the $pr^\alpha_{j,k}$ are variables. Adding an equation

$$pr^\alpha_{j,1} + pr^\alpha_{j,2} + \cdots + pr^\alpha_{j,n} = \lceil pr^\alpha_{j,1} + pr^\alpha_{j,2} + \cdots + pr^\alpha_{j,n} \rceil$$

for every j such that $w_j \models \phi$, will ensure that either $\sum_{w' \in W} R_\alpha(w_j, w') = 1$ or $\sum_{w' \in W} R_\alpha(w_j, w') = 0$, for every $w_j \in C$, as stated in Definition 2.3.

Let $m = |\Omega|$. Let $\Omega^\# = (\varsigma_1, \varsigma_2, \ldots, \varsigma_m)$ be an ordering of the observations in Ω. With each observation in $\varsigma \in \Omega^\#$, we associate a variable pr^ς_j, where j represents the world in which ς is perceived. For every formulae of the form $(\Sigma, \phi \Rightarrow (\varsigma|\alpha) \bowtie q) \in \Gamma$ and $(\Sigma, \phi \Rightarrow \neg(\varsigma|\alpha) \bowtie q) \in \Gamma$, for every j such that $w_j \models \phi$,

$$pr^{\varsigma|\alpha}_j \bowtie q, \text{ respectively, } pr^{\varsigma|\alpha}_j \not\bowtie q$$

is in $SI(\Gamma)$. Adding an equation

$$pr^{\varsigma_1|\alpha}_j + pr^{\varsigma_2|\alpha}_j + \cdots + pr^{\varsigma_m|\alpha}_j = \lceil (pr^\alpha_{1,j} + pr^\alpha_{2,j} + \cdots + pr^\alpha_{n,j})/n \rceil$$

for every j such that $w_j \models \phi$, ensures that for all $w_j \in C$, if there exists a $w_i \in C$ such that $R_\alpha(w_i, w_j) > 0$, then $\sum_{\varsigma \in \Omega} Q_\alpha(w_j, \varsigma) = 1$, else $\sum_{\varsigma \in \Omega} Q_\alpha(w_j, \varsigma) = 0$, as stated in Definition 2.3.

3.2.2 Belief Update

Let $\Pi(e_h, \alpha, \varsigma)$ be the abbreviation for the term

$$\sum_{j=1}^{n} pr^{\varsigma|\alpha}_j \sum_{i=1}^{n} pr^\alpha_{i,j} \omega^{e_h}_i,$$

which is the probability of reaching the belief-state after performing belief update $[\![\alpha + \varsigma]\!]$ at activity point e_h. And let $BT(e_h, k, \alpha, \varsigma)$ be the abbreviation for the term

$$\frac{pr^{\varsigma|\alpha}_k \sum_{i=1}^{n} pr^\alpha_{i,k} \omega^{e_h}_i}{\Pi(e_h, \alpha, \varsigma)},$$

which is the probability of being in world w_k after performing belief update $[\![\alpha + \varsigma]\!]$ at activity point e_h, where $n = |C|$.

Suppose Σ is $0 \xrightarrow{\alpha_0,\varsigma_0} e_1 \xrightarrow{\alpha_1,\varsigma_1} e_2 \cdots \xrightarrow{\alpha_{z-1},\varsigma_{z-1}} e_z$ and $\Sigma \neq 0$. For every formulae of the form $(\Sigma, \Psi) \in \Gamma$, the following equations are in $SI(\Gamma)$.

$$\omega^{e_{h+1}}_k = BT(e_h, k, \alpha_h, \varsigma_h)$$

for $k = 1, 2, \ldots, n$ and $h = 0, 1, \ldots, z-1$,

$$\Pi(e_h, \alpha_h, \varsigma_h) \neq 0$$

for $h = 0, 1, \ldots, z-1$ and

$$\omega^{e_h}_1 + \omega^{e_h}_2 + \cdots + \omega^{e_h}_n = 1$$

for $h = 0, 1, \ldots, z$, where e_0 is 0. Observe that the e_h are integers and we enforce the constraint that $e_i < e_j$ iff $i < j$.

3.2.3 Continuity and Belief Literals

For every formula of the form $(\Sigma e, Cont(\alpha, \varsigma)) \in \Gamma$ or $(\Sigma e, \neg Cont(\alpha, \varsigma)) \in \Gamma$,

$$\Pi(e, \alpha, \varsigma) \neq 0, \text{ respectively, } \Pi(e, \alpha, \varsigma) = 0$$

is in $SI(\Gamma)$.

For every formula of the form $(\Sigma e, \mathbf{B}\varphi \bowtie p) \in \Gamma$,

$$c_1 \omega^e_1 + c_2 \omega^e_2 + \cdots + c_n \omega^e_n \bowtie p,$$

is in $SI(\Gamma)$, where $c_k = 1$ if $w_k \models \varphi$, else $c_k = 0$.

3.2.4 Rewards, Costs and Utilities

For every formula of the form $(\Sigma, \phi \Rightarrow \mathit{Reward}(r)) \in \Gamma$ and $(\Sigma, \phi \Rightarrow \neg\mathit{Reward}(r)) \in \Gamma$, for every j such that $w_j \models \phi$,

$$R_j = r, \text{ respectively, } R_j \neq r$$

is in $\mathit{SI}(\Gamma)$.

For every formula of the form $(\Sigma, \phi \Rightarrow \mathit{Cost}(\alpha, r)) \in \Gamma$ and $(\Sigma, \phi \Rightarrow \neg\mathit{Cost}(\alpha, r)) \in \Gamma$, for every j such that $w_j \models \phi$,

$$C^{\alpha}_j = r, \text{ respectively, } C^{\alpha}_j \neq r$$

is in $\mathit{SI}(\Gamma)$.

Let $RC(\alpha, e) \overset{def}{=} \omega^e_1(R_1 - C^{\alpha}_1) + \omega^e_2(R_2 - C^{\alpha}_2) + \cdots + \omega^e_n(R_n - C^{\alpha}_n)$. For every formula of the form $(\Sigma e, \mathbf{U}\llbracket\alpha\rrbracket \bowtie q) \in \Gamma$,

$$RC(\alpha, e) \bowtie q$$

is in $\mathit{SI}(\Gamma)$.

To keep track of dependencies between variables in inequalities derived from utility literals of the form $(\Sigma, \mathbf{U}\llbracket\alpha\rrbracket\Lambda \bowtie q)$, we define a *utility tree*. A set of utility trees is induced from a set Δ which is defined as follows (examples follow the formal description). For every formula of the form $(\Sigma e, \mathbf{U}\llbracket\alpha\rrbracket\Lambda \bowtie q) \in \Gamma$, let $(e \xrightarrow{\alpha,\varsigma} e^{\varsigma}, \Lambda) \in \Delta$, for every $\varsigma \in \Omega$, where e^{ς} is a fresh integer. Then, for every $(\xi, \llbracket\alpha\rrbracket\Lambda) \in \Delta$ (where Λ is not empty), for every $\varsigma \in \Omega$, if $(\xi', \Psi) \in \Delta$ such that $\xi' = \xi \xrightarrow{\alpha,\varsigma} e^{\varsigma'}$, then $(\xi', \Lambda) \in \Delta$, else $(\xi \xrightarrow{\alpha,\varsigma} e^{\varsigma}, \Lambda) \in \Delta$, where e^{ς} is a fresh integer. This finishes the definition of Δ. The following example should clarify the meaning Δ and utility trees.

Suppose $\Omega = \{\varsigma_1, \varsigma_2\}$ and

$(\Sigma \xrightarrow{\alpha',\varsigma'} 13, \mathbf{U}\llbracket\alpha_5\rrbracket = 88)$,

$(\Sigma \xrightarrow{\alpha',\varsigma'} 13, \mathbf{U}\llbracket\alpha_1\rrbracket\llbracket\alpha_2\rrbracket > 61)$,

$(\Sigma \xrightarrow{\alpha',\varsigma'} 13, \mathbf{U}\llbracket\alpha_1\rrbracket\llbracket\alpha_3\rrbracket\llbracket\alpha_2\rrbracket < 62)$,

$(\Sigma \xrightarrow{\alpha',\varsigma'} 13, \mathbf{U}\llbracket\alpha_1\rrbracket\llbracket\alpha_4\rrbracket = 63)$,

$(\Sigma \xrightarrow{\alpha',\varsigma'} 23, \mathbf{U}\llbracket\alpha_1\rrbracket\llbracket\alpha_2\rrbracket \geq 64)$ and

$(\Sigma \xrightarrow{\alpha',\varsigma'} 23, \mathbf{U}\llbracket\alpha_2\rrbracket\llbracket\alpha_1\rrbracket = 65)$

are in some leaf node Γ'. Then $(\Sigma \xrightarrow{\alpha',\varsigma'} 13, \mathbf{U}\llbracket\alpha_5\rrbracket = 88)$ is not involved in the definition of Δ', nevertheless, $RC(\alpha_5, 13) = 88$ is in $\mathit{SI}(\Gamma')$.

With respect to the other utility literals,

$(13 \xrightarrow{\alpha_1,\varsigma_1} 24, \llbracket\alpha_2\rrbracket)$, $(13 \xrightarrow{\alpha_1,\varsigma_2} 25, \llbracket\alpha_2\rrbracket)$,

$(13 \xrightarrow{\alpha_1,\varsigma_1} 24, \llbracket\alpha_3\rrbracket\llbracket\alpha_2\rrbracket)$, $(13 \xrightarrow{\alpha_1,\varsigma_2} 25, \llbracket\alpha_3\rrbracket\llbracket\alpha_2\rrbracket)$,

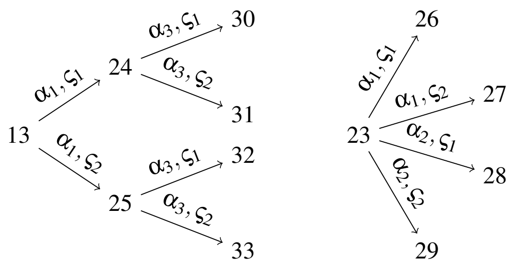

Figure 1: The two utility trees generated from Δ'.

$(13 \xrightarrow{\alpha_1,\varsigma_1} 24, \llbracket\alpha_4\rrbracket)$, $(13 \xrightarrow{\alpha_1,\varsigma_2} 25, \llbracket\alpha_4\rrbracket)$,

$(23 \xrightarrow{\alpha_1,\varsigma_1} 26, \llbracket\alpha_2\rrbracket)$, $(23 \xrightarrow{\alpha_1,\varsigma_2} 27, \llbracket\alpha_2\rrbracket)$,

$(23 \xrightarrow{\alpha_2,\varsigma_1} 28, \llbracket\alpha_1\rrbracket)$ and $(23 \xrightarrow{\alpha_2,\varsigma_2} 29, \llbracket\alpha_1\rrbracket)$

are in Δ'. And due to $(13 \xrightarrow{\alpha_1,\varsigma_1} 24, \llbracket\alpha_3\rrbracket\llbracket\alpha_2\rrbracket), (13 \xrightarrow{\alpha_1,\varsigma_2} 25, \llbracket\alpha_3\rrbracket\llbracket\alpha_2\rrbracket) \in \Delta'$, the following are also in Δ'.

$(13 \xrightarrow{\alpha_1,\varsigma_1} 24 \xrightarrow{\alpha_3,\varsigma_1} 30, \llbracket\alpha_2\rrbracket)$,

$(13 \xrightarrow{\alpha_1,\varsigma_1} 24 \xrightarrow{\alpha_3,\varsigma_2} 31, \llbracket\alpha_2\rrbracket)$,

$(13 \xrightarrow{\alpha_1,\varsigma_2} 25 \xrightarrow{\alpha_3,\varsigma_1} 32, \llbracket\alpha_2\rrbracket)$ and

$(13 \xrightarrow{\alpha_1,\varsigma_2} 25 \xrightarrow{\alpha_3,\varsigma_2} 33, \llbracket\alpha_2\rrbracket)$.

Note how an activity point is represented by the same integer (for instance, 24) if and only if it is reached via the same sequence of actions and observations (for instance, $13 \xrightarrow{\alpha_1,\varsigma_1}$).

The set of utility trees is generated from Δ as follows. Δ is partitioned such that $(e \xrightarrow{\alpha,\varsigma} e', \Lambda)$, $(e'' \xrightarrow{\alpha',\varsigma'} e''', \Lambda') \in \Delta$ are in the same partitioning if and only if $e = e''$. Each partitioning represents a unique utility tree with the first activity point as the root of the tree. For example, one can generate two utility trees from Δ'; one with root 13 and one with root 23. Each activity sequence of the members of Δ represents a (sub)path starting at the root of its corresponding tree. Figure 1 depicts the two utility trees generated from Δ'.

Before considering the general case, we illustrate the method of generating, from the utility trees in Figure 1, the required inequalities which must be in $\mathit{SI}(\Gamma')$.

The formula $(\Sigma \xrightarrow{\alpha',\varsigma'} 13, \mathbf{U}\llbracket\alpha_1\rrbracket\llbracket\alpha_2\rrbracket > 61) \in \Gamma'$ is represented by

$$RC(\alpha_1, 13) + \Pi(13, \alpha_1, \varsigma_1) RC(\alpha_2, 24) + \\ \Pi(13, \alpha_1, \varsigma_2) RC(\alpha_2, 25) > 61$$

in $\mathit{SI}(\Gamma')$. To generate this inequality, the utility tree rooted at 13 is used: See that α_1 is executed at activity point 13, α_2 is executed at activity point 24 if ς_1 is perceived and α_2 is executed at activity point 25 if ς_2 is perceived. Moreover, the latter two rewards must be weighted by the probabilities of reaching the respective new belief-states/activity points.

The formula $(\Sigma \xrightarrow{\alpha',\varsigma'} 13, \mathbf{U}[\![\alpha_1]\!][\![\alpha_4]\!] = 63) \in \Gamma'$ is represented by

$$RC(\alpha_1, 13) + \Pi(13, \alpha_1, \varsigma_1) RC(\alpha_4, 24) + \Pi(13, \alpha_1, \varsigma_2) RC(\alpha_4, 25) = 63.$$

in $SI(\Gamma')$. This time, $\underline{\alpha_4}$ is executed at the activity points 24 and 25.

Next, the utility tree rooted at 23 is used to find the representation of $(\Sigma \xrightarrow{\alpha',\varsigma'} 23, \mathbf{U}[\![\alpha_1]\!][\![\alpha_2]\!] \geq 64) \in \Gamma'$. Looking at the utility tree, one can work out that

$$RC(\alpha_1, 23) + \Pi(23, \alpha_1, \varsigma_1) RC(\alpha_2, 26) + \Pi(23, \alpha_1, \varsigma_2) RC(\alpha_2, 27) \geq 64$$

must be in $SI(\Gamma')$.

For $(\Sigma \xrightarrow{\alpha',\varsigma'} 23, \mathbf{U}[\![\alpha_2]\!][\![\alpha_1]\!] = 65) \in \Gamma'$,

$$RC(\alpha_2, 23) + \Pi(23, \alpha_2, \varsigma_1) RC(\alpha_1, 28) + \Pi(23, \alpha_1, \varsigma_2) RC(\alpha_1, 29) \geq 64$$

is in $SI(\Gamma')$.

Formula

$$(\Sigma \xrightarrow{\alpha',\varsigma'} 13, \mathbf{U}[\![\alpha_1]\!][\![\alpha_3]\!][\![\alpha_2]\!] < 62) \in \Gamma', \qquad (1)$$

is represented by the inequality shown in Figure 2. The size of the utility tree rooted at 13 is due to (1). Hence, the whole tree is employed to generate the inequality.

In general, for every utility literal of the form

$$(\Sigma e_z, \mathbf{U}[\![\alpha_1]\!][\![\alpha_2]\!] \cdots [\![\alpha_y]\!] \bowtie q).$$

in leaf node Γ, an inequality can be generated from an associated utility tree and the inequality must be in $SI(\Gamma)$. We do not have space to go into the details, but please see the thesis (Rens, 2014, Chap. 8) for details.

Theorem 3.1. *The decision procedure is sound, complete and terminating. The SDL is thus decidable with respect to entailment as defined above.*

Proof. Please refer to the thesis (Rens, 2014, Chap. 8) for the proof. □

Although the SDL vocabulary is finite, the need to deal with probabilistic information makes the above decidability result non-trivial.

4 Domain Specification

First we present a framework for domain specification with the logic, then we look at some examples of SDL entailment in use.

4.1 The Framework

The framework presented here should be viewed as providing guidance; the knowledge engineer should adapt the framework as necessary for the particular domain being modeled. On the practical side, in the context of the SDL, the domain of interest can be divided into five parts:

Static laws (denoted as the set *SL*) have the form $\phi \Rightarrow \varphi$, where ϕ and φ are propositional sentences, and ϕ is the condition under which φ is always satisfied. They are the basic laws and facts of the domain. For instance, "A full battery allows me at most four hours of operation", "I sink in liquids" and "The charging station is in sector 14". Such static laws cannot be explicitly stated in traditional POMDPs.

Action rules (denoted as the set *AR*) must be specified. In this paper, we ignore the frame problem (McCarthy and Hayes, 1969); a solution in the current setting requires careful machinery and space prohibits giving it the attention it deserves. We have made preliminary progress in this direction (Rens et al., 2013). For this paper, we identify two kinds of action rules.

The basic kind is the *effect axiom*. For every action α, effect axioms take the form

$$\begin{aligned}
&\phi_1 \Rightarrow [\alpha]\varphi_{11} = p_{11} \land \cdots \land [\alpha]\varphi_{1n} = p_{1n} \\
&\phi_2 \Rightarrow [\alpha]\varphi_{21} = p_{21} \land \cdots \land [\alpha]\varphi_{2n} = p_{2n} \\
&\qquad \vdots \\
&\phi_j \Rightarrow [\alpha]\varphi_{j1} = p_{j1} \land \cdots \land [\alpha]\varphi_{jn} = p_{jn},
\end{aligned}$$

where (i) for every rule i, the sum of transition probabilities $p_{i1}, \ldots, p_{in}$ must lie in the range $[0,1]$ (preferably 1), (ii) for every rule i, for any pair of effects φ_{ik} and $\varphi_{ik'}$, $\varphi_{ik} \land \varphi_{ik'} \equiv \bot$ and (iii) for any pair of conditions ϕ_i and $\phi_{i'}$, $\phi_i \land \phi_{i'} \equiv \bot$.

The knowledge engineer must keep in mind that if the transition probabilities do not sum to 1, the specification is incomplete. Suppose, for instance, that for rule i, $p_{i1} + \cdots + p_{in} < 1$. Then one or more transitions from a ϕ_i-world has not been mentioned and some logical inferences will not be possible.

The second kind of action rule is the *inexecutability axiom*. We shall assume that the set of effect axioms for an action is complete, that is, that the knowledge engineer intends that the conditions of these axioms are the only conditions under which the actions can be executed. Note that $[\alpha]\top > 0$ implies that α is executable. Therefore, if there is an effect axiom for α with condition ϕ, then one can assume the presence of an executability axiom $\phi \Rightarrow [\alpha]\top > 0$. However, we must still specify that an action is inexecutable when none of the effect axiom conditions is met. Hence, the

$$
RC(\alpha_1,13)\begin{array}{l}
+\Pi(13,\alpha_1,\varsigma_1)\left(RC(\alpha_3,24)\begin{array}{ll} +\Pi(24,\alpha_3,\varsigma_1) & RC(\alpha_2,30) \\ +\Pi(24,\alpha_3,\varsigma_2) & RC(\alpha_2,31) \end{array}\right) \\
+\Pi(13,\alpha_1,\varsigma_2)\left(RC(\alpha_3,25)\begin{array}{ll} +\Pi(25,\alpha_3,\varsigma_1) & RC(\alpha_2,32) \\ +\Pi(25,\alpha_3,\varsigma_2) & RC(\alpha_2,33) \end{array}\right)
\end{array} < 62
$$

Figure 2: The inequality representing $(\Sigma \xrightarrow{\alpha',\varsigma'} 13, \mathbf{U}[\![\alpha_1]\!][\![\alpha_3]\!][\![\alpha_2]\!] < 62) \in \Gamma'$.

following *inexecutability* axiom is assumed present.[4]

$$\neg(\phi_1 \vee \cdots \vee \phi_j) \Rightarrow [\alpha]\top = 0$$

where $\phi_1, \ldots, \phi_j$ are the conditions of the effect axioms for α.

Perception rules (denoted as the set PR) must be specified. Let $E(\alpha) = \{\varphi_{11}, \varphi_{12}, \ldots, \varphi_{21}, \varphi_{22}, \ldots, \varphi_{jn}\}$ be the set of all effects of action α executed under all executable conditions. For every action α, perception rules typically take the form

$$
\begin{aligned}
&\phi_1 \Rightarrow (\varsigma_{11} \mid \alpha) = p_{11} \wedge \cdots \wedge (\varsigma_{1m} \mid \alpha) = p_{1m} \\
&\phi_2 \Rightarrow (\varsigma_{21} \mid \alpha) = p_{21} \wedge \cdots \wedge (\varsigma_{2m} \mid \alpha) = p_{2m} \\
&\qquad\vdots \\
&\phi_k \Rightarrow (\varsigma_{k1} \mid \alpha) = p_{k1} \wedge \cdots \wedge (\varsigma_{km} \mid \alpha) = p_{km},
\end{aligned}
$$

where (i) the sum of perception probabilities $p_{i1}, \ldots, p_{im}$ of any rule i must lie in the range $[0,1]$ (preferably 1), (ii) for any pair of conditions ϕ_i and $\phi_{i'}$, $\phi_i \wedge \phi_{i'} \equiv \bot$ and (iii) $\phi_1 \vee \phi_2 \vee \cdots \vee \phi_k \equiv \bigvee_{\varphi \in E(\alpha)} \varphi$. If the sum of perception probabilities $p_{i1}, \ldots, p_{im}$ of any rule i is 1, then any observations not mentioned in rule i are automatically *unperceivable* in a ϕ_i-world. However, in the case that the sum is not 1, this deduction about unperceivability cannot be made. Then the knowledge engineer should keep in mind that a perception rule of the form

$$\phi_i \rightarrow \cdots \wedge (\varsigma \mid \alpha) = 0 \wedge \cdots$$

implies that ς is unperceivable in a ϕ_i-world given that the world is reachable via α. Hence, if $p_{i1} + \cdots + p_{im} \neq 1$ and unperceivability information is available, it should be included with a subformula of the form $(\varsigma \mid \alpha) = 0$.

Utility rules (denoted as the set UR) must be specified. Utility rules typically take the form

$$\phi_1 \Rightarrow Reward(r_1), \quad \ldots, \quad \phi_j \Rightarrow Reward(r_j),$$

meaning that in all worlds where ϕ_i is satisfied, the agent gets r_i units of reward. And for every action α,

$$\phi_1 \Rightarrow Cost(\alpha, r_1), \quad \ldots, \quad \phi_j \Rightarrow Cost(\alpha, r_j),$$

meaning that the cost for performing α in a world where ϕ_i is satisfied is r_i units. The conditions are disjoint as for action and perception rules.

The fifth part of the domain specification is the agent's initial belief-state IB. That is, a specification of the worlds the agent should believe it is in when it becomes active, and probabilities associated with those worlds should be provided. In general, an initial belief-state specification should have the form

$$\mathbf{B}\varphi_1 \bowtie p_1 \quad \wedge \quad \mathbf{B}\varphi_2 \bowtie p_2 \quad \wedge \quad \cdots \quad \wedge \quad \mathbf{B}\varphi_n \bowtie p_n,$$

where (i) $\bowtie \in \{<, \leq, =, \geq, >\}$ and (ii) the φ_i are mutually exclusive propositional sentences (i.e., for all $1 \leq i, j \leq n$ s.t. $i \neq j$, $\varphi_i \wedge \varphi_j \equiv \bot$). For a *full/complete* specification of a *particular* initial belief-state, all the $\bowtie$ must be $=$ and $p_1 + p_2 + \ldots + p_n$ must equal 1.

The union of SL, AR, PR and UR is referred to as an agent's *background knowledge* and is denoted BK. In practical terms, the question to be answered in the SDL is whether $BK \models IB \rightarrow \Theta^-$ holds, where $BK \subset \mathcal{L}_{SDL}$, IB is as described above, and $\Theta^- \in \mathcal{L}_{SDL}^{\not\approx}$ is some sentence of interest, where $\mathcal{L}_{SDL}^{\not\approx}$ is the subset of formulae of $\mathcal{L}_{SDL}$ excluding law literals.

4.2 Examples

This section states three entailment queries based on the oil-drinking scenario. Except for the initial belief-state[5], the following is a full specification of the POMDP model.

Action Rules

$\neg h \Rightarrow [g](f \wedge h) = 0.8 \wedge [g](\neg f \wedge h) = 0.1 \wedge$
$[g](\neg f \wedge \neg h) = 0.1; \quad h \Rightarrow [g]\top = 0.$
$h \Rightarrow [d](\neg f \wedge h) = 0.95 \wedge [d](\neg f \wedge \neg h) = 0.05;$
$\neg h \Rightarrow [d]\top = 0.$
$f \wedge h \Rightarrow [w](f \wedge h) = 1; \quad f \wedge \neg h \Rightarrow [w](f \wedge \neg h) = 1;$
$\neg f \wedge h \Rightarrow [w](\neg f \wedge h) = 1; \quad \neg f \wedge \neg h \Rightarrow [w](\neg f \wedge \neg h) = 1.$

Perception Rules

$\top \Rightarrow (N \mid g) = 1 \wedge (N \mid d) = 1.$
$f \wedge h \Rightarrow (L \mid w) = 0.1 \wedge (M \mid w) = 0.2 \wedge (H \mid w) = 0.7.$
$\neg f \wedge h \Rightarrow (L \mid w) = 0.5 \wedge (M \mid w) = 0.3 \wedge (H \mid w) = 0.2.$
$\neg h \Rightarrow (\forall v^{\varsigma})\neg(v^{\varsigma} = N) \rightarrow (v^{\varsigma} \mid w) = \frac{1}{3}.$

[4]Inexecutability axioms are also called *condition closure* axioms.

[5]Probabilities used for specifying the initial belief-state are assumed given by a knowledge engineer or computed in an earlier process.

Utility Rules

$f \Rightarrow Reward(0)$; $\quad \neg f \wedge h \Rightarrow Reward(10)$;
$\neg f \wedge \neg h \Rightarrow Reward(-5)$.
$\top \Rightarrow (\forall v^{\alpha})(v^{\alpha} = g \vee v^{\alpha} = d) \rightarrow Cost(v^{\alpha}, 1)$;
$f \Rightarrow Cost(w, 2)$; $\quad \neg f \Rightarrow Cost(w, 0.8)$.

The robot gets 10 units of reward for holding the can while it is not full (implying the robot drank the oil), and it gets -5 units of reward for not holding the can while it is not full. Otherwise, the robot gets no rewards. It costs two units to weigh the can when the can is full, else it costs 0.8 units. Grabbing and drinking always costs one unit.

Suppose that the initial belief-state is specified as

$$\mathbf{B}f = 0.7 \wedge \mathbf{B}(\neg f \wedge h) = 0.2 \wedge \mathbf{B}(\neg f \wedge \neg h) = 0.1.$$

Note that it is not fully specified. We determined that BK entails

$$\mathbf{B}f = 0.7 \wedge \mathbf{B}(\neg f \wedge h) = 0.2 \wedge \mathbf{B}(\neg f \wedge \neg h) = 0.1 \rightarrow [\![g + N]\!][\![w + M]\!]\mathbf{B}h > 0.85.$$

That is, the agent's degree of belief that it is holding the can is greater than 0.85 after grabbing the can and then weighing and perceiving that it has medium weight follows from BK, given an initial belief-state $\mathbf{B}f = 0.7 \wedge \cdots = 0.1$. We draw the reader's attention to the fact that sensible entailments can be queried, even with a partially specified initial belief-state.

In the next example, we provide a complete specification of the initial belief-state, but we under-specify the perception probabilities. Suppose that instead of perception rule $f \wedge h \Rightarrow (L \mid w) = 0.1 \wedge (M \mid w) = 0.2 \wedge (H \mid w) = 0.7 \in BK$, we have only $f \wedge h \Rightarrow (H \mid w) = 0.7 \in BK'$. Also assume the perception rule $f \wedge h \Rightarrow (M \mid w) \geq 0.2 \in BK'$. (That is, we modify BK to become BK'.) Then

$$\mathbf{B}(f \wedge h) = 0.35 \ \wedge \ \mathbf{B}(f \wedge \neg h) = 0.35 \ \wedge \ \mathbf{B}(\neg f \wedge h) = 0.2 \ \wedge \ \mathbf{B}(\neg f \wedge \neg h) = 0.1 \rightarrow [\![g + N]\!][\![w + M]\!]\mathbf{B}h > 0.85$$

is entailed by BK'.

Finally, we have shown that BK entails

$$\mathbf{B}f = 0.7 \wedge \mathbf{B}(\neg f \wedge h) = 0.2 \wedge \mathbf{B}(\neg f \wedge \neg h) = 0.1 \rightarrow [\![g + N]\!]\mathbf{U}[\![d]\!][\![d]\!] \leq 7,$$

where, the initial belief-state is under-specified. This example shows that non-trivial entailments about the utility of sequences of actions can be confirmed, even without full knowledge about the initial belief-state.

5 Concluding Remarks

We presented a modal logic with a POMDP semantics for representing stochastic domains and reasoning about noisy actions and observations. Entailment queries can be answered as a solution to certain kinds of projection problems, even with incomplete domain specifications. The procedure for deciding entailment is proved sound, complete and terminating. As a corollary, the entailment question for the SDL is decidable.

Our work can likely be enhanced in several dimensions by further studying the ongoing research in the field of probabilistic logics, stochastic/probabilistic satisfiability, relational (PO)MDPs and symbolic dynamic programming (Saad, 2009; Wang and Khardon, 2010; Sanner and Kersting, 2010; Lison, 2010; Shirazi and Amir, 2011). As espoused by Wang et al. (2008), for instance, there are advantages to being able to model a domain with relational predicates and not only propositions. The SDL thus needs to be lifted to a first-order fragment.

Automatic plan generation is highly desirable in cognitive robotics and for autonomous systems modeled as POMDPs. In future work, we would like to take the SDL as the basis for developing a language or framework with which plans can be generated, in the fashion of DTGolog (Boutilier et al., 2000).

POMDP methods do not deal with the problem of belief maintenance over incomplete models, and this is why the problem is interesting, provided that the solution can lead to methods that are at least, minimally effective. Littman et al. (2001)'s article seems like a good starting point for the investigation to determining the computational complexity of the procedure, our next task.

REFERENCES

Bacchus, F., Halpern, J., and Levesque, H. (1999). Reasoning about noisy sensors and effectors in the situation calculus. *Artificial Intelligence*, 111(1–2):171–208.

Boutilier, C. and Poole, D. (1996). Computing optimal policies for partially observable decision processes using compact representations. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, pages 1168–1175, Menlo Park, CA. AAAI Press.

Boutilier, C., Reiter, R., Soutchanski, M., and Thrun, S. (2000). Decision-theoretic, high-level agent programming in the situation calculus. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-00) and of the Twelfth Conference on Innovative Applications of Artificial Intelligence (IAAI-00)*, pages 355–362. AAAI Press, Menlo Park, CA.

De Weerdt, M., De Boer, F., Van der Hoek, W., and Meyer, J.-J. (1999). Imprecise observations of mobile robots specified by a modal logic. In *Proceedings of the Fifth Annual Conference of the Advanced School for Computing and Imaging (ASCI-99)*, pages 184–190.

Gabaldon, A. and Lakemeyer, G. (2007). $\mathcal{ESP}$: A logic of only-knowing, noisy sensing and acting. In *Proceedings of the Twenty-second National Conference on Artificial Intelligence (AAAI-07)*, pages 974–979. AAAI Press.

Geffner, H. and Bonet, B. (1998). High-level planning and control with incomplete information using POMDPs. In *Proceedings of the Fall AAAI Symposium on Cognitive Robotics*, pages 113–120, Seattle, WA. AAAI Press.

Hansen, E. and Feng, Z. (2000). Dynamic programming for POMDPs using a factored state representation. In *Proceedings of the Fifth Intional Conference on Artificial Intelligence, Planning and Scheduling (AIPS-00)*, pages 130–139.

Hansson, H. and Jonsson, B. (1994). A logic for reasoning about time and reliability. *Formal Aspects of Computing*, 6:512–535.

Iocchi, L., Lukasiewicz, T., Nardi, D., and Rosati, R. (2009). Reasoning about actions with sensing under qualitative and probabilistic uncertainty. *ACM Transactions on Computational Logic*, 10(1):5:1–5:41.

Kwiatkowska, M., Norman, G., and Parker, D. (2010). Advances and challenges of probabilistic model checking. In *Proceedings of the Forty-eighth Annual Allerton Conference on Communication, Control and Computing*, pages 1691–1698. IEEE Press.

Levesque, H. and Lakemeyer, G. (2004). Situations, si! Situation terms no! In *Proceedings of the Conference on Principles of Knowledge Representation and Reasoning (KR-04)*, pages 516–526. AAAI Press.

Lison, P. (2010). Towards relational POMDPs for adaptive dialogue management. In *Proceedings of the ACL 2010 Student Research Workshop*, ACLstudent '10, pages 7–12, Stroudsburg, PA, USA. Association for Computational Linguistics.

Littman, M., Majercik, S., and Pitassi, T. (2001). Stochastic boolean satisfiability. *Journal of Automated Reasoning*, 27(3):251–296.

McCarthy, J. (1963). Situations, actions and causal laws. Technical report, Stanford University.

McCarthy, J. and Hayes, P. (1969). Some philosophical problems from the standpoint of artificial intelligence. *Machine Intelligence*, 4:463–502.

Monahan, G. (1982). A survey of partially observable Markov decision processes: Theory, models, and algorithms. *Management Science*, 28(1):1–16.

Poole, D. (1998). Decision theory, the situation calculus and conditional plans. *Linköping Electronic Articles in Computer and Information Science*, 8(3).

Rens, G. (2014). *Formalisms for Agents Reasoning with Stochastic Actions and Perceptions*. PhD thesis, School of Mathematics, Statistics and Computer Science, University of KwaZulu-Natal.

Rens, G., Meyer, T., and Lakemeyer, G. (2013). On the logical specification of probabilistic transition models. In *Proceedings of the Eleventh International Symposium on Logical Formalizations of Commonsense Reasoning (COMMONSENSE 2013)*, University of Technology, Sydney. UTSe Press.

Rens, G., Meyer, T., and Lakemeyer, G. (2014a). A logic for specifying stochastic actions and observations. In Beierle, C. and Meghini, C., editors, *Proceedings of the Eighth International Symposium on Foundations of Information and Knowledge Systems (FoIKS)*, Lecture Notes in Computer Science, pages 305–323. Springer-Verlag.

Rens, G., Meyer, T., and Lakemeyer, G. (2014b). SLAP: Specification logic of actions with probability. *Journal of Applied Logic*, 12(2):128–150.

Ross, S., Pineau, J., Chaib-draa, B., and Kreitmann, P. (2011). A bayesian approach for learning and planning in partially observable markov decision processes. *J. Mach. Learn. Res.*, 12:1729–1770.

Saad, E. (2009). Probabilistic reasoning by sat solvers. In Sossai, C. and Chemello, G., editors, *Proceedings of the Tenth European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU-09)*, volume 5590 of *Lecture Notes in Computer Science*, pages 663–675, Berlin, Heidelberg. Springer-Verlag.

Sanner, S. and Kersting, K. (2010). Symbolic dynamic programming for first-order POMDPs. In *Proceedings of the Twenty-fourth National Conference on Artificial Intelligence (AAAI-10)*, pages 1140–1146. AAAI Press.

Shirazi, A. and Amir, E. (2011). First-order logical filtering. *Artificial Intelligence*, 175(1):193–219.

Smallwood, R. and Sondik, E. (1973). The optimal control of partially observable Markov processes over a finite horizon. *Operations Research*, 21:1071–1088.

Wang, C., Joshi, S., and Khardon, R. (2008). First order decision diagrams for relational MDPs. *Journal of Artificial Intelligence Research (JAIR)*, 31:431–472.

Wang, C. and Khardon, R. (2010). Relational partially observable MDPs. In Fox, M. and Poole, D., editors, *Proceedings of the Twenty-fourth AAAI Conference on Artificial Intelligence (AAAI-10)*. AAAI Press.

Wang, C. and Schmolze, J. (2005). Planning with POMDPs using a compact, logic-based representation. In *Proceedings of the Seventeenth IEEE International Conference on Tools with Artificial Intelligence (ICTAI-05)*, pages 523–530, Los Alamitos, CA, USA. IEEE Computer Society.